TCP Segment
- IP packet
- TCP packet
- TCP segment

Sequence Number
- Sequence number = first byte
- ACK sequence number = next expected byte

ISN
If a TCP packet is not delivered, later on the initial session is terminated, we can have another session started we wan to prevent sequence number from mixing up.

TCP requires changing the ISN over time

TCP Three Way Handshake
SYN -> SYN ACK (Host B will reserve memory for Host A) -> ACK (can send data at this point, host A does not have to wait for host B)

Both SYN & SYN ACK has a timer, if no signal is received, then retry.
Host A can start to send data right after ACK because Host A has received SYN ACK from Host B previously. Data is exchanged in both directions.

Ack field in the table for B: A’s ISN + 1
: B’s ISN + 1

What if the SYN packet gets lost?
Two reasons of lost packet
1. Packet is lost inside the network (lots of intermediate hops)
2. Server rejects the packet

If the SYN is lost
- User clicks reload to trigger an “abort” of the connect
- Creates another socket

Automatic Repeat reQuest

Reasons for Retransmission
- Early timeout means the timer is too short.
- Fix timer vs. variable timer (better) ←-- the reason why TCP is different from UDP

How long should Sender wait? [Efficient]
TCP sets timeout as a function of the RTT
Smooth estimate: keep a running average of the RTT
est. RTT = a * est. RTT + (1-a) * Sample RTT

A flaw in this approach
1. Sample RTT too long or too short
2. Does not consider variance in the RTT
3. 2RTT for timeout does not work

Solution (not a algorithm)

TCP Sliding Window: allow a larger amount of data in flight

Why?
- Stop and wait is inefficient when delay * bandwidth is high: blind waiting

Fast Retransmission
- Better solution under sliding window
- Have the receiver send ACK packets
- Work best at long data transfer / large window size/ low burstiness in packet losses

Tanner Evans

Sequence Numbers:
- 1st bytes is the sequence number.
- Ack follows the sequence number

Initial Sequence Number:
- Start with a different sequence number for each new TCP connection to increase security and reduce collisions of packets being received with the same seqno

TCP Three-Way Handshake:
- Syn used to open connection and syn ack confirms the connection with sender of syn
- Ack is sent by the send and data can now be sent
- Timers are used on both sides to verify acks are received appropriately

TCP Header:
step1
- syn Ack sends seqno and receiving machine can infer the next seqno to expect.
  Flag Syn, Ack

Step2
- syn-ack sends b’s initial sequence and confirms A’s seqno using acknowledgement field

Step3
- ack a sends b’s seqno using the acknowledgement field to confirm it

SYN Packet Loss:
- Timers take care of knowing if a syn packet has been lost
- It is unknown how long these timers should wait as there is no way to know how far the recipient is

Syn Loss and Web Downloads:
- If a syn is lost during web download it waits 3-6 seconds and sends another syn
  - sometimes users can speed up this process by clicking hyperlink again

Reasons for Retransmission:
- Speed ups in transmission speed mean hardcoded lengths for timers might be unnecessarily long
How long should sender wait?
-RTT time can be estimated by observing ack and update the wait timer accordingly
Flaw:
-Varying sample time could mean drastic variation of RTT time
Motivation for Sliding Window:
-Allows for more efficient use of bandwidth than stop and wait.
Fast Retransmission:
-can be achieved by using acks
Effectiveness of fast retransmit:
-Long data transfers
-Large window
-low burst of packets

Andrew Copley 10/27/2016
Stream bytes to segment
TCP Segment
- IP packet
- TCP packet
- TCP segment
MSS = max segment size

Sequence Number
- Sequence number = first byte
- ACK sequence number = next expected byte

ISN = initial segment number
- practical issue / ports used again & could get a wrong packet.
- security issue / hackers guess ISN & hijack connection.
- ISN changed for TCP over time / 32bit clock ticks every 4 ms.
- hosts need to exchange seq number.

TCP Three Way Handshake
- SYN -> SYN ACK(Host B will reserve memory for Host A) -> ACK, can send data at this point, host A does not have to wait for host B
- Both SYN & SYN ACK has a timer, if no signal is received, then retry.

Host A can piggy back data right after ACK because Host A has received SYN-ACK from Host B previously.

TCP HEADER
- usually 20 bytes, but can vary.
- Flags - SYN, FIN, RST, PSH, URG, ACK

How can SYN packet get lost?
- Packet is lost in network
- Server rejects the packet
If SYN is lost
- User clicks reload to trigger an “abort” of the connection, then another socket is created.

ARQ = Automatic Repeat reQuest

Reasons for Retransmission
- Packet loss
- Ack loss (Duplicate packet)
- Early Timeout (Duplicate packet)

How long should timer be?
- too long or short is inefficient
- timeout set as a function of RTT
- sender knows rtt by time between acks
- Timeout = 2 x estimated RTT

Flaw in timeout approach
- retransmission ambiguity = only collect samples for segments sent 1 time
- Does not consider variance in the RTT
- using 2 x RTT for timeout does not work on high loads, variance is too high
- Solution - Jacobsen/karls (not a algorithm)

TCP Sliding Window: allow more data on the wire at any given time
Receiver buffering
- window size
- receiver advertises window size to sender

Fast Retransmission
- Better solution under sliding window
  Idea - receiver sends acks, ack says still waiting on nth packet